数理計画法のデータベース(数値解析とそのアルゴリズム)

AUTHOR(S):

八巻, 直一; 本郷, 茂; 宮田, 雅智

CITATION:

八巻, 直一 ...[et al]. 数理計画法のデータベース(数値解析とそのアルゴリズム). 数理解析研究所講究録 1992, 791: 175-188

ISSUE DATE:

1992-06

URL:

http://hdl.handle.net/2433/82682

数理解析研究所講究録
第 791 巻 1992 年 175-188

数理計画法のデータベース

Database of mathematical programming methods

システム計画研究所　八巻直一 (Naokazu Yamaki)
専修大学　経営学部　本郷　茂 (Shigeru Hongo)
青山学院女子短期大学　宮田雅智 (Masanori Miyata)

1 はじめに

数理計画法は，近年とくに進展し，どんどん新しいアルゴリズムが開発されつつある．今後共，その勢いは衰えることなく，ますます数多くの有用なアルゴリズムが現れるであろう．

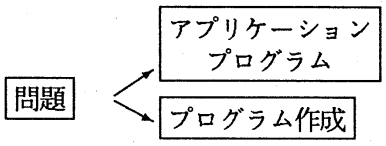
わが国では，戦後どちらかというと OR より IE，すなわち最適化よりも改善が重要視されてきたが，このところ，少しずつ変化が見られ、数理計画法が注目されるようになりつつある．事実，製造業や金融などでは，実際に数理計画法を導入する動きが盛んである．それに対して，数理計画法のソフトウェアはそう多くはなく，実務家自身が書物に記載されたアルゴリズムを，苦労してプログラミングする場合も少なくない．また、研究者は自分の発案によるアルゴリズムの検証や，他の論文のアルゴリズムの追試など，面倒なプログラミングをこなさなければならない．

もし，書物に書かれたアルゴリズムが，そのままデータベースに登録されて，それを検索 (参照) し統合し，自動的に FORTRAN などのソースコードを構築するプロセッサがあれば，アルゴリズムの蓄積と検証，さらには実用化が一層促進されるであろう．

本稿では，このような立場から，アルゴリズムの記述言語とデータベースについて考察する．

2 データベースのアイデア

数理計画に限らず，問題を解きたい場合，アルゴリズムに従って，自らプログラムを作って計算させるか，あるいは，既成の汎用アプリケーションプログラムを使うかのいずれかであろう．汎用のアプリケーションと呼ばれるものの多くは，入力パラメタ解析，問題の解法，計算結果の出力などの処理を行う多くのモジュールプログラムを一体化し，利用者からみるとブラックボックスの形式をとっている場合が多く見られる．この方式は解法の選択やパラメタの設定などがかなり柔軟にでき，一つの問題に対し，解法やパラメタをいろいろ変化させながら処理するには大変便利である．反面，システムに手を加えることは，まず不可能である．また，新しいアルゴリズムを検証したい場合は，コンプリートプログラムを作成して実験する他に手段はない．



そこで，我々は次のようなシステムを提案している．

その利用手順を図式化すると，おおよそ図1のようになる．

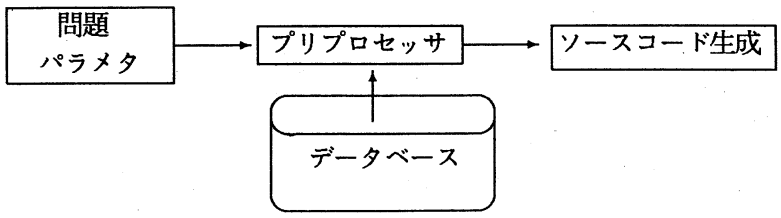


図 1: プログラム作成の流れ

実際には，データベースに蓄積されている部品を集めてモジュールを作り，モジュールを集めてコンプリートプログラムを自動生成する．その様子は図2のようになる。

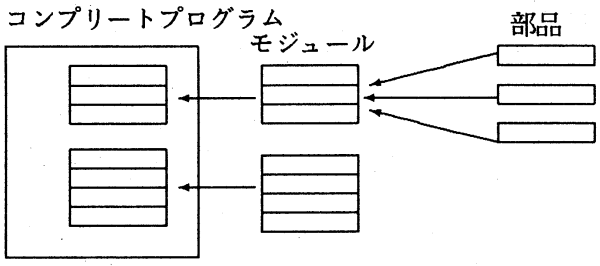


図 2: 部品化ソフトウェアの構造

我々は，上記の処理系を実現するために，以下の二つ言語を開発中である．[1]

ASNOP:: ユーザによって与えられた問題に適した FORTRAN のソースプログラムを生成するプリプロセッサである．最適化手法のプログラムはデータベースに部品として蓄えられており，それらが **ASNOP** によって自動的に組み合わされ，完結した FORTRAN ソースが得られる．

LAMAX-S:: FORTRAN77 に強力な行列操作機能を付加した言語である．行列やベクトルの変数の定義とそれらの演算，さらにバンド行列，スパース行列などの属性の定義，行列を値として返す関数の記述などの機能を持ち，**LAMAX-S** のソースプログラムは，**LAMAX-S** プロセッサによって FORTRAN に変換される．その際，ベクトル化が行われ演算の高速化が図られる．

上の二つのシステムは，それぞれ別の目的に沿って開発されたが，書物に記載された数理計画法のアルゴリズムが，ほとんど行列表現されていることと，アルゴリズムの構造がだいたい決まった表現を持つことから，我々は **ASNOP** と **LAMAX-S** を組み合わせることを考えた．それは，**LAMAX-S** を用いることにより，表記のアルゴリズム記述とデータベースの構築が，容易にできることが期待されるからであ

る．このようなシステムであれば，理論やアルゴリズムの研究者が，新しい方法を一つの部品としてシステムに追加して，比較的容易にそれを実験することができる．

新しいアルゴリズム ⟶ **ASNOP 規則に従い LAMAX-S で記述** ⟶ 部品箱 (データベース) に登録

3 ASNOP の概要

ASNOP は現在すでにリリースされている．ここでは，今回，試みとして追加した新しい機能と **LAMAX-S** 記述によるソースコード生成の概要とデータベースについて述べる．

3.1 ソースコードの生成

データベースには，アルゴリズムの部品が入っている．利用者は，それを抽出し，問題に適したプログラムのソースコードが生成できる．いってみればイージーオーダー式のアプリケーションシステムであるといえる．ソースコードを生成するのは，**ASNOP** プリプロセッサによる．したがって，**ASNOP** プリプロセッサは，アルゴリズム・データベースの参照用のツールと位置づけられる．プリプロセッサは，従来，ソースコード生成の機能として以下の機能を持っていた．

- 部品の選択と抽出
  データベースから必要な部品を抜取りパッチワーク的に結合する．
- 文番号の自動生成
  結合したソースコードの文番号が FORTRAN 規則に違反しないように自動生成する．
- コメントの処理
  インライン・コメントの実現とコメントをドキュメントとして抜き取る．

本研究は，現在の版に改良を加えてアルゴリズムの記述性および，データベース機能の強化を図った．

図 1で示した，問題とパラメタ記述から実行までの流れを改めて図式化すると図 3のようになる．

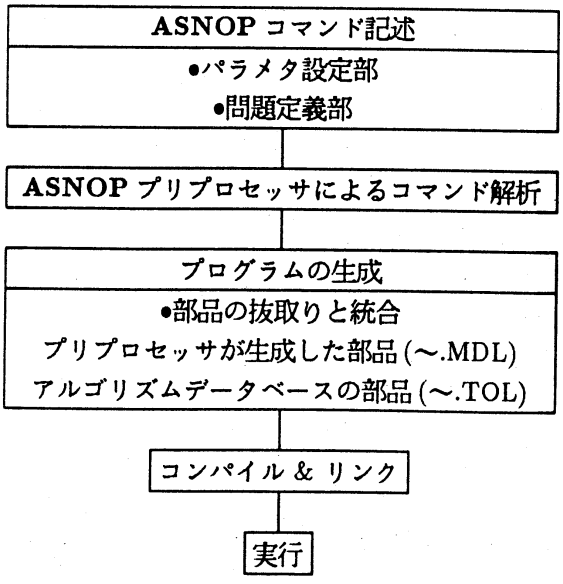


図 3: 処理の流れ

ASNOP の基本的機能は変わっていない．改良点は，以下の機能である．

- 一時部品としてユーザ記述部分の部品化
  問題に依存する記述 (例えば，変数の数) を部品化する．

- step ラベルからの文番号生成
  書物などによく出てくる，アルゴリズムの step 記述から，FORTRAN の文番号を生成する．

3.2 データベース参照機能

ここでは，データベース参照機能として，**ASNOP** プリプロセッサが持っているコマンドをあげておく．プリプロセッサは，内容により 2 つの記述部に分類される．

- パラメタ設定部

- 問題定義部

パラメタ設定部は非線形最適化問題を解く上で必要となる各種のパラメタ (解法，変数次元，制約式の数など) を設定する記述部であり，問題定義部は，目的関数，制約関数などを FORTRAN 言語，**LAMAX-S** あるいはシステムが用意した **ASNOP** コマンド拡張記述で記述する部である．パラメタ設定部のキーワードとパラメタ値を以下に説明する．

MODEL-DEFINITION ;
　　パラメタ設定部の開始宣言である．
TITLE=title ;
　　問題に対する表題を与える．
DIMENSION=ndim ;
　　問題に必要な変数の数を与える．
PROBLEM= { NLP | NLS } ;
　　ASNOP は，非線形最適化問題を非線形計画問題 (NLP) と非線形最小 2 乗問題 (NLS) の 2 つに分類している．
SAMPLE-SIZE=nsamp ;
　　非線形最小 2 乗問題の場合，観測データを入力する必要がある．そのデータ数をこのキーワードで与える．
METHOD= { SQP | AGL | NLSSQP } ;
　　解法として逐次 2 次計画法 (SQP)，拡張ラグランジュ乗数法 (AGL)，最小 2 乗問題用 SQP(NLSSQP) が準備されており，そのいずれを選択するかを指示する．
INEQUALITY-CONSTRAINT=nieq ;
　　不等号制約式の数を与える．
EQUALITY-CONSTRAINT=neq ;
　　等号制約式の数を与える．
INNER-ITERATION-LIMIT=linner ;
　　拡張ラグランジュ乗数法の中で解かれる無制約最小化の部分に要する反復回数の上限値を与える．
OUTER-ITERATION-LIMIT=louter ;
　　拡張ラグランジュ乗数法の反復回数あるいは SQP 法と NLSSQP の反復回数の上限値を与える．
EVAL-ITERATION-LIMIT=leval ;
　　目的関数の評価回数の上限値を与える．
EPS-TOLERANCE=eps ;
　　最適解に達したか否かの収束判定基準を与える．
DELTA-TOLERANCE=delta ;
　　導関数値を差分近似で求めるときのきざみ幅を与える．
ZERO-TOLERANCE=zero ;
　　検査対象たとえば hj(x) が | hj(x) | ≦ zero の時は 0 と見なすときの基準値を与える．
END-MODEL-DEFINITION ; パラメタ設定部の終了宣言である．

ASNOP システムに与える情報として，前述でしたパラメタ設定部では表現できないもの，すなわち非線形最適化問題の目的関数，制約式あるいは初期点などは，問題定義部で記述する．記述された内容は，一時的な部品として生成され，データベース参照の対象となる．この記述部は次のような 11 種類がある．

```
USER-PROGRAM-DESCRIPTION   ;   宣言文
INITIAL-PROCESS   ;   初期プロセス
POST-PROCESS   ;   終了プロセス
OBJECTIVE-FUNCTION   ;   目的関数
ELEMENT-FUNCTIONS   ;   最小２乗要素関数
INEQUALITY-CONSTRAINT-FUNCTIONS   ;   不等号制約関数
EQUALITY-CONSTRAINT-FUNCTIONS   ;   等号制約関数
OBJECTIVE-FIRST-DERIVATIVE-FUNCTIONS   ;   目的関数の１階偏導関数
ELEMENT-FIRST-DERIVATIVE-FUNCTIONS   ;   最小２乗要素関数の１階偏導関数
INEQUALITY-CONSTRAINT-FIRST-DERIVATIVE-FUNCTIONS   ;   不等号制約関数の１階偏導関数
EQUALITY-CONSTRAINT-FIRST-DERIVATIVE-FUNCTIONS   ;   等号制約関数の１階偏導関数
```

3.3 アルゴリズム部品

ASNOP の中核部分は非線形最適化のためのデータベースであり，現在リリースしている版のデータベースには，以下のアルゴリズムが入っている．

- AGL : Augmented Lagrangian Multiplier Method
- SQP : Sequential Quadratic Programming Method
- NLSSQP : SQP Method for Nonlinear Least Squares Problems

そして，その部品は解法で分類すると，次のように大きく３つのフレームに分かれる．

非線形最適化問題

- SQP フレーム
  - 等号制約条件付き SQP
  - 不等号制約条件付き SQP
  - 等号・不等号制約条件付き SQP
  - 無制約 SQP
- AGL フレーム
  - 等号制約条件付き AGL
  - 不等号制約条件付き AGL
  - 等号・不等号制約条件付き AGL
  - 無制約 AGL

非線形最小２乗問題

- NLS フレーム
  - 等号制約条件付き NLS
  - 不等号制約条件付き NLS
  - 等号・不等号制約条件付き NLS
  - 無制約 NLS

この他には，導関数値を差分近似で求める部品や結果の標準出力を行う部品などがある．また，部品は，その性質上次の2つの部品群に分けられている．

1. **ASNOP** プリプロセッサ生成部品

2. アルゴリズムデータベース部品

ASNOP プリプロセッサが生成する部品は，処理の都度，毎回作成されるものである．すなわち，問題に依存していて，あらかじめ部品としてデータベース化できない記述を一時的に部品化するものである．

アルゴリズムデータベース部品は，最適化手法のアルゴリズム部品として蓄えられたものである．先に述べた3つのフレーム及び標準出力部品などは，アルゴリズムデータベース部品に属している．このアルゴリズムデータベースには，標準部品が蓄えられているが，**ASNOP** システムの利用者が自分の発案によるアルゴリズムの検証や，書物に記載された数理計画法のアルゴリズムにそくした部品を新たに追加したり，標準部品を変更したりすることもできる．次に，部品の例をあげておく．

1. **ASNOP** プリプロセッサ生成部品名

```
パラメタ設定部
    %PARAM.MDL    :  パラメタ設定部から生成されるパラメタ文
                     MODEL-DEFINITION;
問題定義部
    %DESCR.MDL    :  宣言文 (USER-PROGRAM-DESCRIPTION;)
    %INITP.MDL    :  初期プロセス (INITIAL-PROCESS;)
    %POSTP.MDL    :  終了プロセス (POST-PROCESS;)
    %OBJF.MDL     :  目的関数 (OBJECTIVE-FUNCTION;)
    %EOBJF.MDL    :  最小2乗要素関数 (ELEMENT-FUNCTIONS;)
    %ICONST.MDL   :  不等号制約条件式 (INEQUALITY-CONSTRAINT-FUNCTIONS;)
    %ECONST.MDL   :  等号制約条件式 (EQUALITY-CONSTRAINT-FUNCTIONS;)
    %DOBJF.MDL    :  目的関数の1階偏導関数
                       (OBJECTIVE-FIRST-DERIVATIVE-FUNCTIONS;)
    %DEOBJF.MDL   :  最小2乗要素関数の1階偏導関数
                       (ELEMENT-FIRST-DERIVATIVE-FUNCTIONS;)
    %DICONST.MDL  :  不等号制約条件式の1階偏導関数
                       (INEQUALITY-CONSTRAINT-FIRST-DERIVATIVE-FUNCTIONS;)
    %DECONST.MDL  :  等号制約条件式の1階偏導関数
                       (EQUALITY-CONSTRAINT-FIRST-DERIVATIVE-FUNCTIONS;)
```

2. アルゴリズムデータベースの現在の標準部品

```
骨格部品 (7個)
      FRMAIN.TOL    : 最初に呼び出すフレーム
      LMXEMAIN.TOL  : ＬＡＭＡＸ－Ｓフレーム
      FOREMAIN.TOL  : ＦＯＲＴＲＡＮフレーム
      FOREFRM.TOL   : ＡＳＮＯＰメインフレーム
      PARAMCOM.TOL  : システム変数のＣＯＭＭＯＮ
      PARAMVAR.TOL  : システム変数の初期値設定
                                        etc.

拡張ラグランジュ乗数法 (12個)
      AGLF.TOL      : 関数評価回数のカウントアップ
      AGLLINE.TOL   : 直線探索
      AGLMULTI.TOL  : 乗数とペナルティ・パラメタの更新
      AGUPDATE.TOL  : 近似行列Ｈ (ヘッセ行列の逆行列) の更新
      CD.TOL        : 探索方向の計算
                                        etc.

逐次2次計画法 (11個)
```

```
        PENALTY.TOL   : ペナルティ関数の計算
        SQPLINE.TOL   : 直線探索
        SQPMULTI.TOL  : ラグランジュ乗数の計算
        SQUPDATE.TOL  : 行列更新
        UPR1K1.TOL    : ペナルティ・パラメタの更新
                                    etc.

最小 2 乗問題 (2 個)
       CAM.TOL        : 行列 A の計算
       CCM.TOL        : 行列 C の計算

共通部品 (31 個)
       DCONST.TOL     : 制約の導関数の計算
       DOBJF.TOL      : 目的関数の導関数の計算
       QP.TOL         : QP 部分問題の計算
       FRREPORT.TOL   : 拡張機能 C%Report により呼び出される
                                    etc.
```

4 LAMAX-S の概要

LAMAX-S(LAnguage for MAtriX calculation - Super computer) は、FORTRAN 言語に行列演算の機能を取り入れた言語でである．すなわち，その概念は以下のとうりである．

$$\textbf{LAMAX} - \textbf{S} = FORTRAN + \text{行列演算機能} - \text{若干の機能}$$

一口に行列演算機能と言っても，単に加算や乗算だけでなく，**LAMAX-S** では，連立方程式を解いたり，ＬＵ分解したりするための機能も備えている．FORTRAN のすべての機能はそのまま使え，拡張された行列機能はなるべく FORTRAN 言語と違和感のないように考慮されている．

4.1 行列演算

演算は，FORTRAN と同じように，数式通りに記述できる．たとえば，

(1) $$B = (X'X)^{-1}X'y$$

という式は，**LAMAX-S** では，

```
B = (X'*X)**(-1)*X'*y
```

と記述する．連立一次方程式の解法を用いるならば，**LAMAX-S** では solve 文を用いる．

```
solve (X'*X) * B = X'*y
```

この方法で解くと，逆行列を求めるよりも少ない計算量で解を求めることができる．**LAMAX-S** は，このように行列演算という高度に抽象化された段階で，最適化を行うことができるという利点を備えている．

表 1: **LAMAX-S** の行列属性

構造	要素の密度	対称性
矩形行列 (m×n)		
正方行列		
列ベクトル	密行列	対称行列
行ベクトル		
バンド行列	スパース行列	非対称行列
対角行列		
3 重対角行列		
上三角行列		
下三角行列		

4.2 各種構造のサポート

LAMAX-S は，矩形行列だけでなく，さまざまな構造をした行列を扱うことができる．たとえば，バンド行列や三角行列，対角行列，スパースおよび，対称などである．**LAMAX-S** では，これらを**行列属性**という概念で処理する．行列属性は，次の表 1 に示すように，**構造・要素の密度・対称性**に分類される．下線の付いたものが既定値となりる．

行列構造は，型宣言文で宣言する．たとえば，次の宣言文は，10×5 の**矩形行列** Q を宣言する．要素の型は整数型である．

```
integer Q:rectangular[10,5]
```

次の宣言文は，複素数型の要素を 20 個持つ**列ベクトル (column vector)** V を宣言する．

```
complex V:vector[20]
```

これらの宣言は，要素の型が同じであれば，一つの文の中にまとめて記述することが可能である．

```
real*8 x,Y:rectangular[5,10],Z:vector[5],W:rvector[10]
```

x は，行列ではなく，FORTRAN の通常の倍精度実数型の変数である．上記の宣言で，rvector という構造名があるが，これは**行ベクトル (row vector)** を意味する．

5 アルゴリズム記述とデータベース機能の実際

先に述べたように，本研究では **ASNOP** に 2 つの新しい機能を追加した．ここでは，まず，非線形最適化アルゴリズム部品と利用者の問題記述を **LAMAX-S** の表現でどのように記述するかを示す．そして，機能追加した **ASNOP** プリプロセッサで解析し，ソースコード生成の様子を例示する．その手順は図 4のようになる．

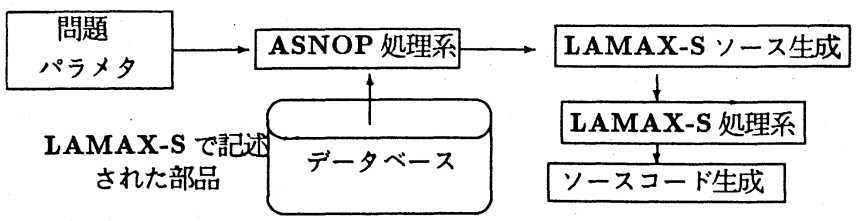


図 4: 新しい処理系

まず，部品の記述例を示す．以下に示す部品はアルゴリズム・データベースに登録されているニュートン法の一部である．
なお，とくに，

```
      real*8 Df  :vector[*]
      real*8 DDf :square[$ndim]
      d = - DDf**(-1) * g     ← (-1) は逆行列を求めている
```

は **LAMAX-S** の記述であり，

```
$         INCLUDE:'%POSTP.MDL' : ・終了プロセス
```

は部品の中でさらに部品を組み込む手続きを示している．

```
C++++Begin File= NEWTON.TOL : ニュートン法
        ........
      real*8 Df  :vector[*]
      real*8 DDf :square[$ndim]
        ........
Step_1 停止条件を検査する．
      fv = f(x)
      g  = Df(x , DDf )
      norm = absmax( g )
      write(*,#100) k,fv,norm
#100  format( '  step=',i4,'   f=',d8.3,'  norm g=',d8.3)
      if( norm .lt. eps ) then
$        INCLUDE:'%POSTP.MDL' : ・終了プロセス
         stop
      endif
Step_2 探索方向 d を求める．
      d = - DDf**(-1) * g
Step_3 d 方向の刻み幅を求める（直線探索）
      fv=ff
      s = size * d
Step_4 k = k + 1 として Step_1 へ行く．
      k = k + 1
      go to Step_1
```

上記のアルゴリズムを **ASNOP** プリプロセッサで処理すると，次のような **LAMAX-S** ソースプログラムを生成する（アルゴリズムの全体は次の章を参照されたい）．実際は，このソースを **LAMAX-S** で処理し，最終的なソースコードを生成するが，冗長になるのでここでは割愛する．

アルゴリズム記述と生成されたソースの文番号の対応は次のようになっている．

```
          STEP_1       →  70010  CONTINUE
          #100         →  70020
          GOTO STEP_1  →  GOTO 70010

*Step_1 停止条件を検査する.
70010    continue
       fv = f(x)
       g  = Df(x , DDf )
       norm = absmax( g )
       write(*,70020) k,fv,norm
70020   format( '  step=',i4,'   f=',d8.3,'  norm g=',d8.3)
       if( norm .lt. eps ) then
*Begin %POSTP.MDL    : POST-PROCESS;
*        FORTRANプログラム(終了プロセス)
         write(*,*) ' f=',fv
         write(*,*) ' x='
         call mprint( x )
*End   %POSTP.MDL    : POST-PROCESS;
         stop
       endif
 *Step_2 探索方向 d を求める.
70030    continue
        d = - DDf**(-1) * g
 *Step_3 d 方向の刻み幅を求める(直線探索)
70040    continue
 c
       fv=ff
       s = size * d
*Step_4 k = k + 1 として Step_1 へ行く.
70050    continue
        k = k + 1
*       go to Step_1
       go to  70010
```

以上の様な形で部品を作った後，利用者が記述したパラメタ設定部，問題定義部を参照し，ソースコードを生成する．以下にPOWELLの問題をニュートン法で解く記述例の一部を示す．

```
      model-definition;
        title= Powell Function (Unconst. Optimiz. Prob.);
        dimension=4;
        method=NEWTON;
         ........
*  ・初期プロセス
      initial-process;
        x = | 3.d00 , -1.d00 , 0.d00 , 0.1d00 |'
      end-initial-process;
         ........
```

title，dimension，method，などのコマンドで与えられた値は，一時的に作成されるPARAM.MDL部品として登録される．

初期プロセスは，計算の初期点を与える手続きで，一時部品INITP.MDLを生成する．ここは**LAMAX-S**の記述になっている．「|3.d00 ,…|」はベクトルを意味しており，さらに「' 」は転置ベクトルであることを意味している．

6 結論と考察

実験の結果，**ASNOP** をプラットホームとして使うと，

- アルゴリズムの追加が容易
- 追加による副作用が無い

また，**LAMAX-S** を使うと

- 行列表現能力が強力なので，アルゴリズム記述が容易

などの結果が得られ，既存の **ASNOP** の機能を害することなく，記述性が向上し，データベースの構築が可能であることを確認した．今後は，さらにユーザーインターフェイス環境を研究して，数理計画ソフトの一つの実現法として確立させたいと考えている．

7 例題

powell の問題をニュートン法で解く過程を示す．

```
%lang lamax-s ;
*モデル記述
      model-definition;
        title= Powell Function (Unconst. Optimiz. Prob.);
        dimension=4;
        method=NEWTON;
        line-search=BISECTZ;
        outer-iteration-limit=100;
        eps-tolerance=1.0d-6;
        zero-tolerance=0.1d-7;
      end-model-definition;
*  ・宣言文
      user-program-description;
*        ＦＯＲＴＲＡＮプログラムの宣言文
        real*8 xa,xb,xc,xd
      end-user-program-description;
*  ・初期プロセス
      initial-process;
*        ＦＯＲＴＲＡＮプログラム（初期プロセス）
        x = | 3.d00 , -1.d00 , 0.d00 , 0.1d00 |'
             end-initial-process;
*  ・終了プロセス
      post-process;
*        ＦＯＲＴＲＡＮプログラム（終了プロセス）
         write(*,*) ' f=',fv
         write(*,*) ' x='
         call mprint( x )
      end-post-process;
*  ・目的関数
      objective-function;
*        ＦＯＲＴＲＡＮプログラム（目的関数）
        xa = x[1] + 10.0d0*x[2]
        xb = x[3] - x[4]
        xc = x[2] - 2.0d0*x[3]
        xd = x[1] - x[4]
```

```
        f  = xa*xa + 5.0d0*xb*xb + xc**4 + 10.0d0*xd**4
      end-objective-function;
*  ・目的関数の１階偏導関数
      objective-first-derivative-functions;
*        ＦＯＲＴＲＡＮプログラム（目的関数の１階偏導関数）
        xa = x[1] + 10.0d0*x[2]
        xb = x[3] - x[4]
        xc = x[2] - 2.0d0*x[3]
        xd = x[1] - x[4]
        Df[1]=  2.0d0*xa + 40.0d0*xd**3
        Df[2]= 20.0d0*xa +  4.0d0*xc**3
        Df[3]= 10.0d0*xb -  8.0d0*xc**3
        Df[4]=-10.0d0*xb - 40.0d0*xd**3
         ........
      end-objective-first-derivative-functions;
```

上の利用者記述を **ASNOP** プリプロセッサで処理すると，まず，部品 FRMAIN が呼び出され，パラメタ設定部で記述された内容に従って，順々にソースが生成される．

各行の先頭文字が$ である行は，ソースコードを生成する際に用いる制御文である．

最初に呼ばれるメインフレーム

```
C++++Begin File= FRMAIN.TOL : 最初に呼ばれるメインフレーム
/* 最初に呼ばれるメインフレーム
$  SET   !TOOLBOX-VER=VER.p3 REV.0!
$   !METHOD=QNEWTON!
$      INCLUDE:'QNEWTON.TOL' : 準ニュートン法
$!
$   !METHOD=NEWTON!
$      INCLUDE:'NEWTON.TOL' : ニュートン法
$!
$   !METHOD=DGW!
$      INCLUDE:'DGW.TOL' : DGW 法
$!
$   !METHOD=SQP+
$   !METHOD=NLSSQP!
$      INCLUDE:'FRSQP.TOL' : 逐次２次計画法
$!
$   !METHOD=AGL+
$   !METHOD=NLSAGL!
$      INCLUDE:'FRAGL.TOL' : 拡張ラグランジュ乗数法
$!
/* C++++End   File= FRMAIN.TOL
```

ニュートン法部品

```
C++++Begin File= NEWTON.TOL : ニュートン法
      PROGRAM ASNOP
$     INCLUDE:'%PARAM.MDL' : モデル記述
       real*8 x   :vector[$ndim]
       real*8 d   :vector[$ndim]
       real*8 s   :vector[$ndim]
       real*8 y   :vector[$ndim]
       real*8 g   :vector[$ndim]
       real*8 w   :vector[$ndim]
       real*8 f , fv
       real*8 alpha , r1 , r2 , gd , ff , sy , yHy , norm
       real*8 Df  :vector[*]
```

```
       real*8 DDf :square[$ndim]
$  !DESCR=YES!
$     INCLUDE:'%DESCR.MDL' : ・宣言文
$  !
$     INCLUDE:'PARAMVAR.TOL' : システム変数の初期値設定
Step_0 初期点 x . k = 0 とおく.
$      INCLUDE:'%INITP.MDL' : ・初期プロセス
       k = 0
Step_1 停止条件を検査する.
       fv = f(x)
       g  = Df(x , DDf )
       norm = absmax( g )
       write(*,#100) k,fv,norm
#100   format( '  step=',i4,'   f=',d8.3,'  norm g=',d8.3)
       if( norm .lt. eps ) then
$         INCLUDE:'%POSTP.MDL' : ・終了プロセス
          stop
       endif
Step_2 探索方向 d を求める.
       d = - DDf**(-1) * g
Step_3 d 方向の刻み幅を求める (直線探索)
$      !LINE-SEARCH=BISECTZ!
$          INCLUDE:'BISECTZ.TOL'
$!
$      !LINE-SEARCH=ARMIJO!
$          INCLUDE:'ARMIJO.TOL'
$!
c
       fv=ff
       s = size * d
Step_4 k = k + 1 として Step_1 へ行く.
       k = k + 1
       go to Step_1
       end
       real*8 function f(x)
$      INCLUDE:'%PARAM.MDL'
       real*8 x:vector[$ndim]
$  !DESCR=YES!
$     INCLUDE:'%DESCR.MDL' : ・宣言文
$  !
$     INCLUDE:'%OBJF.MDL' : ・目的関数
       end
       real*8 :vector[*] function Df(x , DDf )
$      INCLUDE:'%PARAM.MDL'
       real*8 x:vector[$ndim]
       real*8 DDf:square[$ndim]
$  !DESCR=YES!
$     INCLUDE:'%DESCR.MDL' : ・宣言文
$  !
        allocate Df:vector[$ndim]
$     INCLUDE:'%DOBJF.MDL' : ・目的関数の偏導関数
       end
/* C++++End   File= NEWTON.TOL
```

謝辞 この原稿を作る際、TEX の原稿として清書するにあたっては, (株) システム計画研究所 齊藤智也氏の多大な協力を得た. ここに感謝の意を表したい.

参考文献

[1] 非線形最適化プログラミング, **ASNOP** 研究会, 日刊工業新聞社, 1991

[2] 内田, 八巻, 本郷, 唐津,「行列演算用言語 **LAMAX-S** とそのアルゴリズム記述性」, 1991 日本 OR 学会秋期研究発表会アブストラクト集, 138–139